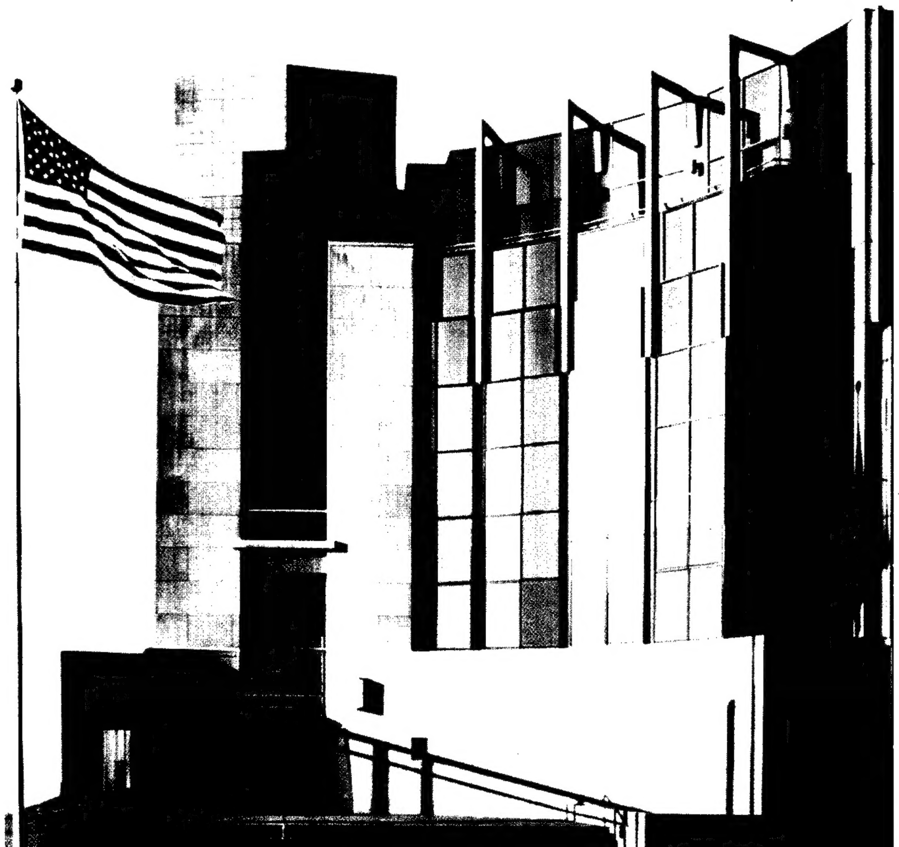
Carnegie Mellon
Software Engineering Institute

Control Channel Toolkit: A Software Product Line Case Study

Paul Clements
Sholom Cohen
Patrick Donohoe
Linda Northrop

September 2001



TECHNICAL REPORT
CMU/SEI-2001-TR-030
ESC-TR-2001-030

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Control Channel Toolkit: A Software Product Line Case Study

CMU/SEI-2001-TR-030
ESC-TR-2001-030

Paul Clements
Sholom Cohen
Patrick Donohoe
Linda Northrop

September 2001

Product Line Practice Initiative

This report was prepared for the National Reconnaissance Office.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Table of Contents

List of Figures

List of Tables

Abstract

This report is a case study of the Control Channel Toolkit (CCT), a software asset base for a software product line of ground-based spacecraft command and control systems built under the direction of the United States National Reconnaissance Office (NRO). Beginning with a characterization of the CCT context and a narration of the history of the effort, the report describes the management and software engineering practices, the software artifacts that were developed, the results that were achieved, and the lessons that were learned. It concludes with an accounting of the measurable benefits the government has already reaped in the initial use of CCT on a specific spacecraft command and control system. With the permission of Addison-Wesley, this report is extracted from *Software Product Lines: Practices and Patterns* [Clements 01], where it was published with the approval of the National Reconnaissance Office.

1 Introduction

A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [Clements 01]. Many organizations are finding that software product lines take economic advantage of the commonality among similar systems and can yield remarkable quantitative improvements in productivity, time to market or field, product quality, and customer satisfaction.

The United States National Reconnaissance Office (NRO) recognized that much is the same from one ground-based spacecraft command and control system to another. They decided to take advantage of the commonality and build a product line asset base for their ground-based spacecraft command and control software. Early results show that they made a fine decision. The first system in their product line is enjoying, among other benefits, 50% reductions in overall cost and schedule, and nearly ten-fold reductions in development personnel and defects. These are impressive results for a first foray into software product lines, and much can be learned from their experience.

This case story is the story of that experience—the story of the Control Channel Toolkit (CCT), the software asset base for a software product line of ground-based spacecraft command and control systems built under the direction of the NRO. We begin by characterizing the CCT context and then we chronicle the history of the effort, describing the justification for CCT, the CCT team's organizational structure and influences, the software artifacts that were developed, and the practices and processes that were used. We also describe the results that were achieved and the lessons that were learned, including the measurable benefits the government has already reaped in the initial use of CCT on a specific spacecraft command and control system.

Throughout, we underscore software product line practice area coverage and patterns as well as how the CCT effort uniquely addressed many of the practice areas. The practice areas and patterns we refer to are thoroughly described in *Software Product Lines: Practices and Patterns* [Clements 01]. With the permission of Addison-Wesley, this report is extracted from this same book, where it was published with the approval of the National Reconnaissance Office.

2 Contextual Background

We begin with an overview sketch of CCT. CCT is a software asset base commissioned by one organization (the NRO) and built under contract by another (the Raytheon Company). This acquirer/contractor arrangement adds a bit of a twist to some of the software product line practice areas and also introduces some interesting economic motivations. Raytheon hopes to use the CCT experience to secure future spacecraft command and control system business. The NRO hopes to persuade other government spacecraft ground system projects to subscribe to the product line effort that it commissioned, thus defraying some of the CCT development and sustainment costs.

CCT is neither a complete software system nor a complete software product line; it is a software product line asset base. The asset base consists of generalized requirements, domain specifications, a software architecture, a set of reusable software components, test procedures, a development environment definition, and a guide for reusing the architecture and components. CCT users build products. CCT users are individual government contractors commissioned by a government office to build spacecraft command and control systems using the CCT software assets. CCT is currently being used to field several government systems. Those systems constitute the actual product line. The NRO is currently funding Raytheon to maintain the CCT for current and future users. Figure 1 annotates our software product line signature diagram [Clements 01] to illustrate.

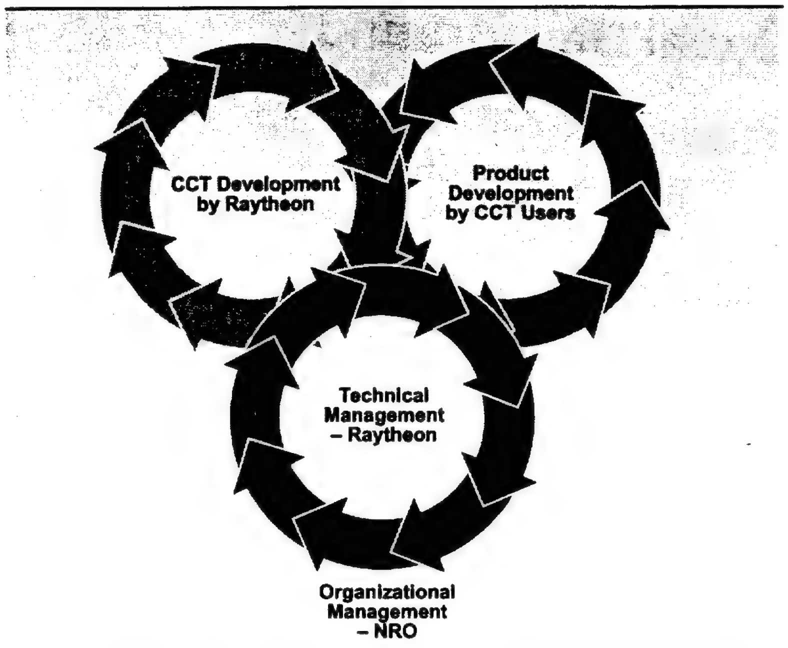

CCT Development by Raytheon
Product Development by CCT Users
Technical Management – Raytheon
Organizational Management – NRO

Figure 1: The Three Essential Activities and CCT

You can see from this illustration that in this case the product development and the core asset development are split among potentially different contractor organizations, and Raytheon, the NRO, and CCT users share the management responsibilities. If we are to understand how this all works smoothly, we need to know something about the two key organizations—the NRO and Raytheon.

2.1 Organizational Profiles

The NRO designs, builds, and operates defense reconnaissance satellites. NRO intelligence products, provided to an expanding list of customers such as the U.S. Central Intelligence Agency (CIA) and the U.S. Department of Defense (DoD), can warn of potential trouble spots around the world, help plan military operations, and monitor the environment. As part of the 13-member Intelligence Community, the NRO plays a primary role in achieving information superiority for the U.S. Government and Armed Forces. The NRO is staffed by both DoD and CIA personnel. It is funded through the National Reconnaissance Program, part of the National Foreign Intelligence Program. The U.S. Assistant Secretary of the Air Force for Space also serves as the Director of the NRO.

In the past, the very existence of the NRO, let alone its software projects, was classified information. In recent years, the NRO has implemented a series of actions declassifying some of its operations. The organization's existence was formally declassified in September 1992, followed by the location of its headquarters in Chantilly, VA, in 1994. In February 1995, CORONA, a photoreconnaissance program in operation from 1960 to 1972, was declassified, and 800,000 CORONA images were transferred to the National Archives and Records Administration. The declassification of the satellite generation following CORONA is currently planned. In December 1996, the NRO announced for the first time, in advance, the launch of a reconnaissance satellite. This new policy of openness is, among other things, intended to increase the NRO's customer and contractor base.[1]

Shrinking budgets have brought all this about. As the NRO director recently remarked,

> *I look back at former NRO Directors, having had tons of money and very little oversight, and they achieved great results. I look at them with great envy because now I have an awful lot of oversight and not nearly as much money. . .Today, virtually all of the imagery that is collected by the National Reconnaissance Office is collected [as] non-sensitive information. It's delivered to military forces around the world. . . [T]he large majority of NRO information can be handled at lower classification levels, opening up the opportunities for partnership. Finally, we continue to find ways of strengthening our partnerships with industry. As the commercial satellite industry matures, we're looking at industrial practices and trying to bring them into our major acquisitions.*[2]

This new atmosphere at the NRO—shrinking budgets, openly looking for customers of NRO products, exploring partnerships with industry—laid the groundwork for turning to software product lines as a cost-saving system acquisition strategy.

The NRO selected Hughes Electronics Corporation as prime contractor for the CCT development. The development team at Hughes had broad experience in satellite ground control systems. This experience was to serve them well when it came time to craft common requirements that were generic enough to serve the first three different satellite projects, yet specific enough to be useful to each of them. They were already knowledgeable in areas such as orbit dynamics, coordinate system transformations, vehicle configuration variability, and other technical aspects that would need to be mastered before they could build any single satellite ground control system, let alone an asset base for an entire family of systems. Furthermore, the Hughes team was accustomed to following defined processes for both software

[1] Source: http://www.nro.gov.

[2] From remarks by Keith Hall, Director of the National Reconnaissance Office, to the National Space Symposium, Colorado Springs, Colorado, 9 April 1998. Source: http://www.nro.odci.gov/speeches/nss-498.html.

project management and software development. The process discipline required for software product line efforts was not a stretch for them.

Hughes Electronics subsequently merged with the Raytheon Company. The consolidation of Hughes into Raytheon also brought together related operations formerly called Raytheon E-Systems, Raytheon Electronic Systems, and the Texas Instruments divisions earlier acquired by Raytheon. Overall, the Raytheon Company operates in both the defense and commercial sectors in the areas of electronics, aircraft systems, and engineering and construction. Raytheon is one of the largest American defense contractors, with $20 billion in sales and more than 92,000 people. Their products include missiles, radar systems, sensors and electro-optics, reconnaissance, surveillance, air traffic control, and aircraft integration systems.[1]

When Hughes was merged with Raytheon, CCT became a Raytheon project, although the development team remained unchanged. There were 20 to 25 employees on the Raytheon development team that took CCT from domain analysis through its architecture creation and implementation. The team peaked at 45 people during component engineering. With the exception of a very small number of subcontractors, all technical work was done at the contractor's site in Denver, Colorado.

2.2 Project History

The Control Channel Toolkit project actually emerged from earlier efforts to produce common software for satellite command and control. In 1993, an Air Force program office began the acquisition of a replacement command and control system for a classified satellite program. The contract for the Distributed Command and Control System (DCCS) began in 1994. The DCCS was designed to fly a specific type of space vehicle. Its design was for the most part object-oriented with some wrapped legacy code that provides highly specialized numerical algorithms.

In 1996, the Air Force Space and Missile Center (SMC) began an effort for a Standard Spacecraft Control Segment (SSCS) that brokered several other satellite program office requirements into a single specification. The SSCS goal was to exploit DCCS to produce a common core capability for use by these other program offices, thus offering development and maintenance savings for SMC. The SSCS technical approach was to modify the DCCS design and code artifacts to meet new (common) requirements. Shortly after development began, it was recognized that the approach was faulty: DCCS had not been designed to address the new requirements. In short, DCCS had not been designed for reuse. All major SSCS

1 Source: 1999 Raytheon annual report, http://www.raytheon.com/finance/1999/ray_annual.pdf.

schedule milestones were missed. The effort was redirected to work on common human machine interfaces as a way of standardizing business processes.

At this point, the Air Force and the NRO met to determine a common development path for SSCS and DCCS that would offer reduced cost and effort in both development and maintenance phases. There was yet another government project, which we will call the Spacecraft C2 System, that had also planned on modifying the DCCS artifacts to meet their requirements. This project was early in its requirements phase. The Air Force requested a revised program plan for all three systems to determine how they could realistically take advantage of the commonalities found in all three and potentially others.

A two-month feasibility study was approved and commenced in May 1997. Six working groups were formed to explore architecture definition, process definition, operations and maintenance definition, cost analysis, schedule assessment, and risk assessment. Learning from the flawed reuse approach of DCCS/SSCS, the NRO initiated the Control Channel Toolkit (CCT) project to reengineer DCCS from an architecture perspective. The Spacecraft C2 Program became the first user of CCT. The Software Engineering Institute (SEI) became involved with the CCT effort at the request of the NRO in October 1997 and continued the collaboration until CCT completion.

CCT was originally conceived to be a "toolkit" that would consist of:

- a set of reusable software components
- tools to help integrate them into complete systems

The toolkit concept evolved into something much more sophisticated and much more usable—namely, a software product line asset base. While a product line asset base certainly includes components and tools, we have seen (and so did the CCT team) that much more is required to achieve strategic, predictable, and measurably beneficial reuse.

2.3 Control Channels

Command and control software is employed throughout the life cycle of a space vehicle project. Typically, each satellite project develops and maintains a separate software system for the purpose of commanding and controlling the spacecraft vehicle. As the complexity of satellite systems has increased over the years, so have the requirements of the ground systems necessary to command and control them. Typical command and control systems exceed 500,000 lines of code.

The CCT supports a product line of ground-based spacecraft command and control systems (also referred to as "control channels"). Satellite control channels provide ground processing support to spacecraft, allowing operations staff to monitor spacecraft functions, configure spacecraft service and payload systems, manage spacecraft orbits and attitudes, and perform mission planning. Control channels receive clear (unencrypted) telemetry data from the front-end processing equipment and release spacecraft and ground system commands. Within a control channel, telemetry data are used to assess spacecraft health and perform payload functions. The telemetry may be provided to external clients as well; these clients may in turn provide command data to the control channel. Control channels also exercise control of the ground antenna system, uplink transmitter operations, and downlink receiver control. These monitoring and command capabilities combine to allow real-time control of the spacecraft and its payload.

The execution of these real-time functions requires connectivity between the control channel software functions and the spacecraft. The period of time and system configuration used to achieve connectivity describe a *contact*. Contacts normally include a preplanned set of tasks, such as assessing state of health, commanding the spacecraft into a new configuration, or executing an orbit or attitude maneuver. Since contacts require line-of-sight visibility from a ground antenna to the spacecraft and use ground resources as well, they are scheduled events determined by the scheduling activities of a spacecraft mission. Unplanned contacts may also be required to perform anomaly resolution for the spacecraft.

Control channels perform orbit and attitude maintenance through processing of tracking and telemetry data and generation of orbit estimates and maneuver data. These data are used to determine the need for future contacts and command parameters as well as the times required to execute the maneuvers. Special consideration is given to launch and early-orbit operations, which often place the spacecraft into unique configurations or impose unique requirements.

Some control channel processing is real-time or near-real-time and is called *execution* processing. The remainder is batch or off-line and is called *planning*. During a contact, the control channel primarily executes capabilities associated with the receipt of telemetry and the release of command data to the ground equipment and spacecraft. The control channel receives raw telemetry from the spacecraft and the antenna ground system through its front-end processing equipment and converts it into client-usable form. The control channel distributes the telemetry to client processes, often with graphical user interfaces (GUIs) provided for operators to make real-time assessments of the spacecraft's health. Client processes may also initiate command requests that cause formatting and transmission of command data to destination services such as the spacecraft or to ground equipment such as the antenna. In many systems, traditional operator functions are automated through the use of special client processes that use rule-based processing.

Control channels archive telemetry and command data as well as other systems data such as operator actions and tracking data. These data form key inputs to the planning functions and provide a historical log for analysis and bookkeeping. Historical data are available for trend analysis and anomaly resolution.

Modern spacecraft use on-board processors to control both platform bus and mission payload functions. Control channels model the on-board processor instruction and data loading and its execution in order to anticipate spacecraft state and behavior.

In off-line processing, the control channel performs planning functions to estimate and propagate the spacecraft orbit and attitude, calculate maneuvers to change orbit or attitude, and schedule future contacts and resource needs. These functions include both launch and early-orbit support for the spacecraft as well as on-orbit operations.

3 Launching CCT

The primary motivation for adopting a product line approach is economic—the promise of long-term savings in return for some extra short-term effort. Thus, business issues often dominate any discussion about a particular product line. The resolution and handling of those business issues may determine the success or failure of the product line effort, as measured by its long-term costs and savings. The feasibility study team, which was already steeped in the understanding of the domain we just summarized, analyzed both business and technical issues. In getting started, they and the initial CCT team that followed exercised practices in the following software product line practice areas:

- Developing a Business Case
- Developing an Acquisition Strategy
- Funding
- Structuring the Organization
- Technical Planning
- Organizational Planning
- Operations

And it was the combination of practices from these areas that accomplished the "Launching" part of the "Launching and Institutionalizing" practice area.

3.1 Developing a Business Case for CCT

The business goals for CCT were to reduce life-cycle costs and development risks, promote interoperability, and provide flexibility. For the government spacecraft community, flexibility translates into accommodating multiple implementation contractors and enabling the integration of both commercially available and legacy assets. The strategy was to develop a toolkit once, and have the three candidate programs share the development and maintenance costs.

The study team examined an array of government and commercial spacecraft command and control systems, including the three under primary investigation. They objectively examined the life-cycle cost data of the DCCS, the SSCS, and the Spacecraft C2 Program. Original program estimates were used; the validity of these estimates was not challenged, although the estimates were considered by the study team to be lower than the actual costs. Based on a domain requirements analysis, cost estimates were derived for implementing the three candidate systems using a product line approach. The commonality of requirements across these systems ranged from a low of 49% to a high of 89%, and the cost-benefit analysis showed significant life-cycle savings. The team also conducted a thorough risk identification and analysis activity. Mitigation strategies were defined for identified risks.

The feasibility study concluded that:

- There was a credible CCT development schedule that contained a supportable staffing plan and no staffing impacts to DCCS and that was timely enough to meet Spacecraft C2 needs.
- CCT would ensure significant cost savings for these three programs over a ten-year period.
- Risks that could preclude CCT's success were manageable.
- CCT supported the overall government reuse vision.

The business case for CCT, as expressed in the original CCT statement of work, described cost savings over multiple programs that were partially attributable to maximized reuse of DCCS assets and exploitation of DCCS experience. Interestingly, the cost analysis did not show major savings for CCT's three candidate systems in terms of development; program sponsors were sold on the vision of greatly reduced risk and the promise of cost savings in the future. The concept of recruiting additional subscribers to the product line—that is, finding new satellite programs that could be built using CCT—was always a key part of the business strategy behind CCT.

3.2 Developing the Acquisition Strategy and Funding CCT

Typically a command and control system is acquired as part of a larger procurement that includes whatever is being commanded and controlled. The contract for the entire procurement is usually awarded to a single prime contractor. In the case of CCT, three government programs formed a partnership to acquire software that could be used across those three programs, as well as others. The CCT acquisition strategy required that the government assume some responsibility development, because spacecraft software systems were becoming too complex to be acquired as stand-alone, one-of-a-kind systems. Otherwise, both the cost and the risk were too great.

Asset ownership was a key product line issue, because the product line was commissioned by one organization but developed by another. In the CelsiusTech example described by Brownsword and Clements, a defense contractor developed a product line on its own initiative and successfully sold it to many governments around the world [Brownsword 96]. In that case, CelsiusTech made sure to retain all intellectual rights to the product line.

In the case of CCT, however, the product line was commissioned by the U.S. Government, which wanted to recover its up-front investment in building the assets by realizing savings across other programs, some of which had not yet been identified. Therefore, it retained the rights to the asset base. In both the CCT and CelsiusTech cases, intellectual rights resided with the organization that (1) paid the up-front cost to build the asset base and (2) took it upon itself to find more customers for the product line to recoup that cost.

Although the NRO commissioned Raytheon for development of product line assets, the assets may be used by other government organizations. These organizations will commission contractors to build specific spacecraft command and control systems. The U.S. Government retains total rights to the CCT architecture and the components. However, Raytheon has full freedom to develop commercialized derivations that they can then bring to the marketplace.

3.3 Structuring the CCT Organization

There are four organizational groups or stakeholders in the CCT effort; the NRO (the sponsoring customer) and Raytheon (the commissioned developer) have already been mentioned. The third set of stakeholders consists of the organizations charged with developing the systems for which the product line was launched: the Spacecraft C2 System, DCCS, and SSCS. The fourth set of stakeholders includes future CCT users—those organizations that may subscribe to CCT as a basis for building or reengineering their own systems in the future. Only the organizational units that the NRO and Raytheon structured for CCT were important to the initial CCT effort.

Raytheon's CCT Program was divided into the organizational units shown in Table 1 to accomplish the CCT development tasks.

Table 1: *Raytheon's Organizational Structure for CCT Development*

Organizational Unit	Responsibilities
Contractor program office	Technical management of development effort; accountable to the NRO CCT Program Office
Program support	Technical management support functions—for example, configuration management, quality assurance, business operations
Domain engineering and architecture	Requirements engineering; architecture definition
Component engineering	Component development
Application engineering	Testing architecture that demonstrates the ability to build products from CCT
Test engineering	Component and assembly testing
Training	Training materials and internal training of CCT personnel
Sustainment engineering	Fixing and enhancing CCT baseline; working with application engineers and users to refine, deliver, and maintain the CCT assets

CCT development was structured into six overlapping increments with an integrated development team responsible for each. The members of the integrated teams came from a cross section of the organizational structure listed in Table 1 with the exception of the first increment, which involved only the Domain Engineering and Architecture group because this increment was primarily dedicated to scoping the CCT product line. The phases and the integrated teams make explicit the iteration we talk about as inherent in core asset development.

The NRO's organizational unit for CCT was the CCT Program Office, which consisted of a small team of government management and technical personnel who managed CCT development at the organizational level. The CCT Program Office was augmented by a small number of technical personnel from two government research and development centers, the SEI and the Aerospace Corporation.

Every product line effort is dependent on strong communication among the stakeholders [Clements 01]. To ensure effective communication and to guide CCT's development over the extended life cycle—creation, usage in an initial product set, usage in a future product set, and evolution—the CCT Program Office created a variety of working groups. The main CCT working group coordinated project-level decisions for the CCT program and ran the stakeholder working groups. In turn, the stakeholder working groups provided direction for the CCT component and sustainment engineering groups. An architecture group advised the main working group on issues relating to the CCT architecture.

Raytheon's integrated teams and the NRO's working groups were a unique aspect of the CCT organizational structure that permitted productive cross-pollination and communication among the stakeholders.

3.4 Organizational and Technical Planning

The NRO and Raytheon partnered in the planning of CCT. CCT development was scheduled to begin in August 1997 and end on December 31, 1999. Plans were developed that spanned this time period. These plans addressed carefully the organizational management needs of the CCT Program Office for reviews and incremental deliveries, Raytheon's technical management needs for plotting and tracking the development processes, and Spacecraft C2's needs for requisite CCT assets for their system development effort. In addition, configuration management, risk management, and quality assurance plans were developed. There was also a transition plan to be used by developers of CCT-based systems, which describes the process for delivering and installing the CCT assets for use by developers of target systems. The formal process of maintaining the CCT assets was documented in a CCT Sustainment Plan.

All of these plans were accessible at the CCT Web site, and all were updated as the CCT effort progressed to depict the current status of the CCT development and integration process. The CCT effort scored high in the two planning practice areas; the development of the plans, the plans themselves, and their practice of religiously keeping the plans visible, accessible, and up-to-date make CCT an exemplar in these practice areas.

3.5 Operations

Operations for core asset development were governed by the plans set in motion. The integrated teams for each of the six development increments carried out the development tasks. The systematic reuse approach adopted by the CCT effort was process-driven and took advantage of what was available—legacy assets, proven algorithms, and commercial off-the-shelf (COTS) products. Value-adding operational processes included frequent technical interchange meetings, weekly status meetings with the CCT program office, and use of the

CCT Web site to post all artifacts and documentation. The government designated more than 20 documents as deliverables in addition to supporting documents created to aid developers and users. Baselined versions of all deliverable documents and links to the CCT models and software development folders were available on the CCT Web site. The site also hosted the CCT master schedule, archives of reviews, and project status information.

The integrated teams and the working groups promoted sufficient community and stakeholder involvement to make traditional long, drawn-out government reviews unnecessary. The CCT Program Office established a modest number of review points at which increment deliverables and the status of development and technical management activities were reviewed. Metrics were defined and the corresponding data collected to track and evaluate progress. A proof-of-concept prototype was developed to show how long it would take to mine assets and integrate them with the architecture.

As for how the CCT effort would connect with the software product line it was to support, the NRO developed the CCT Program Concept of Operations, which documented operations for coordination with CCT users (those organizations developing products from the CCT assets). This concept of operation contained:

- the strategies, tactics, policies, and constraints that describe processes to be used to field products
- the organizations, activities, and interactions involved in fielding the products
- the specific processes, in overview fashion, that provide a process model for fielding a product in terms of when and in what order these processes take place, including dependencies and concurrencies

Also defined are three stages for using CCT to support the envisioned spacecraft command and control product line: asset development, asset sustainment and process refinement, and product line sustainment and improvement.

1. **Asset development**: Development activities, performed by the CCT contractor, translate the set of common user command and control requirements into software components that provide the needed capabilities. The six developmental increments of CCT that were already described were all part of asset development.

2. **Asset sustainment and process refinement**: Delivery and installation of CCT assets as increments are completed by the CCT Contractor to CCT users, including assistance with installing and transitioning these increments into the user's development environment.

3. **Product line sustainment and improvement**: Coordination of continuous improvement of CCT and its attached processes. As use of CCT increases, it is expected that members of the CCT user community will participate in a CCT users group. The sustainment effort will be coordinated by means of the chief CCT working group processes and procedures.

The CCT Concept of Operations called for a Senior Management Panel to provide "corporate oversight" of the product line and to guide its evolution by setting strategic goals (such as attracting particular new customers). The CCT Program Office was to take the lead in evaluating new subscribers. The sustainment costs were to be shared equally by all future partners, although it is easy to imagine variations on this allocation should all parties agree.

The *Business Plan: Asset Development Phase*, an addendum to the concept of operations, established procedures and criteria for working with potential CCT users. These procedures were designed to support evaluation of the ability of the CCT assets to accommodate the technical and programmatic requirements of new users. Also identified were organizational responsibilities, coordination processes, and timelines for conducting these evaluations and establishing agreements with new users. Specifically, the business plan addressed the qualification of new subscribers. A preliminary assessment to evaluate a prospective new program that inquires about using the CCT product line assets was defined. This assessment was designed to determine the answers to the following questions:

- Does the candidate user fit into the CCT vision?
- Will the candidate user's needs alter the scope of the CCT domain?
- Is the candidate user's concept of system operation compatible with the CCT architecture?
- Does the CCT schedule satisfy the candidate user's needs?
- Would there be a potentially significant impact on existing users if the candidate user were to be added to the CCT user community?
- Does the CCT asset base benefit from accommodating the candidate user?
- Are there any significant contracting issues associated with supporting the candidate user?

If the preliminary assessment was favorable, then a detailed evaluation would take place, based on the following criteria:

- Technical
  - extent to which CCT currently satisfies candidate user's requirements
  - benefits to CCT from enhancing capabilities to encompass a greater portion of candidate user's requirements set
  - fraction of candidate user's proposed requirements that fall outside the scope of CCT assets (and user must pursue independently)
  - potential for technology infusion from user to CCT assets and expected functionality improvements
  - compatibility of CCT architecture with candidate user's system concept of operations, including computing platforms, development tools, and operations skill mix and staffing profiles

- Schedule
  - feasibility of CCT program office meeting candidate user's schedule requirements
  - impact of candidate user's requirements on schedule commitments to existing CCT users
- Cost
  - feasibility of identifying and allocating costs for added development work required by candidate user
  - acceptability of cost sharing provisions to candidate user and existing users
  - potential for overall cost savings
- Risk (technical performance, cost, and schedule)
  - level of risk for meeting commitments to candidate user
  - increase in level of risk associated with meeting commitments to existing users
- Contract issues
  - extent to which accommodating the needs of candidate user falls within defined CCT contract scope
  - issues regarding sole-source vs. competitive selection for any additional required development work

CCT also developed a concept of operations for the sustainment phase that contains:

- the strategies, tactics, policies, and constraints that describe how the product line assets will be sustained and used to field future products
- the organizations, activities, and interactions that describe who will participate in sustaining the product line and what these stakeholders do in that process
- the specific operational processes, in overview fashion, that the CCT program will apply to sustain the assets and maintain CCT user collaboration.

The concept of operations documents developed for CCT provide rich examples for organizations that will not be the product developers of the products in their product lines.

4 Engineering the CCT Core Assets

CCT development consisted of six major software engineering processes:

1. Domain engineering and architecture
2. Component engineering
3. Application engineering
4. Test engineering
5. Sustainment engineering
6. Training

These processes incorporate many of the practices of Jacobson's incremental and iterative reuse-based approach [Jacobson 97] and Kruchten's "4+1 view" model of architecture [Kruchten 95], complemented by the test engineering process. As we have already noted, the engineering processes proceeded through six increments, delivering successively more complete versions of CCT assets. These processes map to the product line practice areas, but we will be faithful to CCT terminology for our description.

The domain engineering and architecture processes were responsible for defining and specifying the product line and creating the product line architecture.[1] The component engineering process created the components of the toolkit. Application engineering was an internal reuse process designed to create a spacecraft command and control application for testing, deriving the application from the reusable assets delivered by the other processes. In particular, the application engineering team created demonstration architectures from the CCT assets and, with the test engineering group, created detailed test procedures for validating the reusability of the asset base after each increment was complete. The sustainment engineering process managed the maintenance and evolution of the CCT assets once they were baselined following delivery during each CCT increment.

1 In CCT parlance, domain engineering is distinguished from architecture creation, whereas elsewhere the creation of an architecture is regarded as being part of domain engineering. In this narrative, the process activities relating to the definition and specification of the domain will be described as domain analysis and the architecture-creation activities will be treated under the heading of software architecture. In addition, the term "reference architecture" was used in the CCT effort to describe what we have called the product line architecture.

The domain specification, reference architecture, and components addressed the common core functionality for systems in the product line. Together with the architecture model and Reuse Guide aimed at users of CCT assets, they comprised the major deliverables of the CCT project. The processes for creating and validating assets, the process outputs, and the process interactions evolved during the course of development. An example of this evolution was the addition of an architecture evaluation step to the architecture-creation process, and the creation of the architecture model as a separate deliverable. To a large extent, the documents produced during the CCT creation were a reflection of the processes.

4.1 Domain Analysis

Although a formal domain analysis method was not used, the essential tasks of domain analysis were performed. They:

- captured and analyzed common requirements and their variation across several systems
- synthesized them into a set of common requirements for the product line
- captured the essential terminology

The CCT domain analysis began with an analysis of the requirements of the three satellite command and control systems for which CCT was built: DCCS, SSCS, and the Spacecraft C2 System. This activity defined the scope of the product line and created a set of generalized common requirements for CCT. The final phase of the domain analysis created a specification of satellite ground-based command and control requirements, applying a use-case-driven approach to describe the commonality and variability across the product line.

The domain definition process classified product line requirements into the two domains: execution (the real-time or near-real-time activities that occur during a contact with a satellite) and planning (the non-real-time activities that occur before or after a contact). Within these two domains, the process identified categories of components to be provided by the toolkit (for example, handling of telemetry streams and orbit estimation) and the common services that support these categories (for example, persistent storage services and event notification services). This partitioning of the problem domain provided the basis for the logical view of the CCT architecture: a planning part and an execution part. Raytheon partitioned the requirements on the basis of the consensus of the domain analysts and the contractor's own experience in building previous spacecraft command and control systems.

The domain definition process produced three documents that described the scope and general requirements of the product line:

1. The **Domain Definition Document** provided an overview of the product line. It described the common features of related systems and defined the scope of the CCT assets for supporting those systems. A glossary of product line terminology was also provided. The scope of CCT was defined in terms of the mission and system characteristics that CCT supports, organized in terms of component categories. For example, CCT will support up to two simultaneous telemetry streams per spacecraft, but users have to provide their own persistence and archiving solution since CCT does not provide a standard database solution.
2. The **Generalized Requirements Specification** documented the contractual requirements for CCT. To create this document, requirements from three different customer bases were examined: DCCS, the Spacecraft C2 System, and SSCS. The document captured common capabilities to be provided by the toolkit. Each requirement was briefly described and was classified as being in one of the domains identified by the domain definition process—execution or planning—or categorized among the common services. Variability in the common requirements was expressed in terms of variation points that were more fully described in the Domain Specification Document. This initial requirements process was supported using Rational's Slate tool to create and maintain a matrix of requirements and textual descriptions mapped to unique identifiers and to the component categories. To maintain traceability from the generalized requirements back to their origins in the three analyzed systems, an additional document was created using Slate.
3. The **CCT Domain Specification Document** contained use-case diagrams and descriptions for the common requirements. Variability was incorporated into the use cases as explicit variation points. This document also provided the design for each use case in terms of sequence diagrams that show the interactions among the blocks participating in the use case. It provided the top-level analysis model to be used by the subsequent domain engineering, component engineering, and application engineering processes.

4.2 Architecture

Prior to initiating the architecture activity, the architecture group conducted a survey of architecture techniques to provide early input to the architecture-creation process. They decided to follow the "4+1 view" model of Philippe Kruchten [Kruchten 95] and to use the Unified Modeling Language (UML) to represent it. The process for defining the software architecture evolved considerably during the development of the CCT assets; they were a bit late in getting "architecture religion." During the early increments, the architecture group considered the component categories as fully embodying the architecture. As their understanding matured, the group developed and maintained a CCT architecture model. They also agreed to two architecture evaluations—one for each of the execution and planning logical entities.

4.2.1 Architecture Definition

There were two key documents that described the CCT architecture and its use:

1. The **CCT Architecture Model** provided a complete description of the architecture. This model provided the philosophical underpinnings of the approach, architectural drivers and concepts, and architectural and design patterns.[1] It also included the architectural views: the logical view which consists of the static object model; the development view which showed layers; the dynamic view, which consisted of the component interaction model; and the process view, which consisted of the process interaction model. UML, as supported by the Rational Rose tool, was used.[2]

2. The **CCT Reuse Guide** documented the CCT assets that could be used to build a product in the product line. The audience for this document was software engineers responsible for selecting and utilizing CCT assets when building a product. The Reuse Guide helped compare capabilities of CCT with program requirements for a new command and control system. This was to assist product builders plan their use of CCT assets and determine modifications and extensions that would be needed. For extensions at variation points, the Reuse Guide provided and described the implementing mechanisms. The CCT Reuse Guide was the production plan for building products in the product line.

Architecture definition and component engineering were closely coupled. The architecture imposed constraints on the design of the components and their interactions. In particular, the less-is-more approach of successful architecture, which seeks to limit the number of different design elements and the permissible interaction mechanisms, was used. The CCT architecture definition process addressed these issues by explicitly listing permissible component interaction mechanisms (such as publish-subscribe, callback, event notification, and so on) in the Architecture Model. The model discussed their rationale and the constraints of these mechanisms as part of the description of the reference architecture concepts. The Reuse Guide provided the implementation view that explicitly identified the programming approaches associated with architectural components.

The Common Object Request Broker Architecture (CORBA) was chosen as the basis for the architecture's intercomponent communication infrastructure. Mission-unique modification and feature extension could happen in several ways. Product builders could replace components at the architectural level by wrapping the new (or legacy) components with Interface Description Language (IDL) specifications and using the Object Request Broker (ORB) to integrate them into the runtime system. They could also use ORB common services to add

1 A notable feature of CCT architecture is an abundant use of architectural and design patterns, which provided a vocabulary for describing and reasoning about the architecture.

2 The tool didn't support the capture of additional CCT architectural representations. Layer diagrams, use-case maps, and overall execution and planning architecture diagrams were created, but were captured on the CCT Web site pages and other documents delivered to the NRO.

their components to the architecture transparently by using common events or data. Finally, they could extend CCT component implementations at designated variation points using inheritance or parameters [Hollander 99].

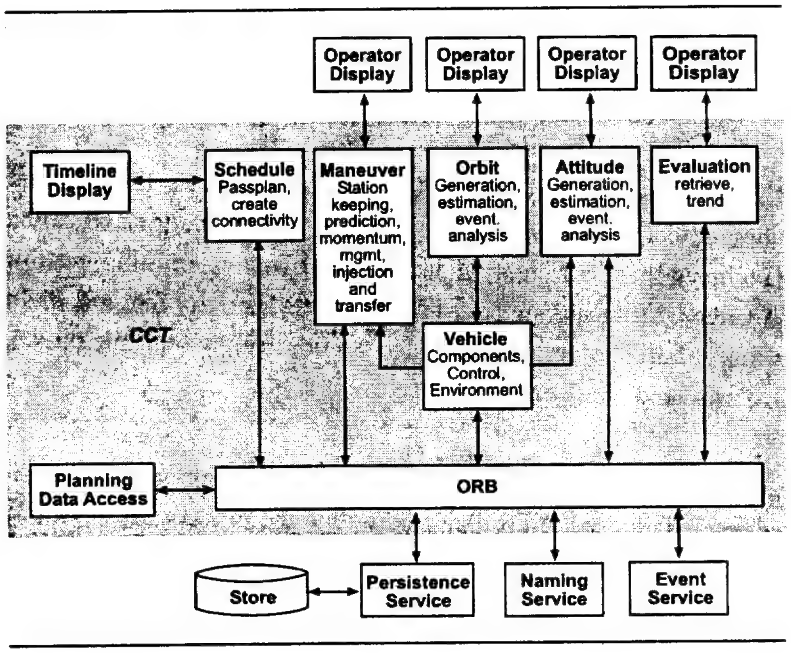
Common services corresponded to CORBA services and facilities. These included naming and event services, object persistence services, and event notification and callback services. Other services included those for creating data-driven displays, printing, manipulating and translating time values, manipulating coordinate systems, activity logging, and managing application event loops. Services are also provided for controlling multiple threads in an application. CCT also provided wrappers for device drivers.

The CCT architecture organized software components into component categories; a category supported development of a specific subsystem within the product line. Within each component category lived related components that users might integrate together to achieve higher-order functionality. Components in different categories might use each other in well-defined ways, so that many system functions would be accomplished through the operating and interaction of components across different categories.

There were two primary subsystems in the CCT architecture: planning and execution. The planning architecture referred to the non-time-critical component categories that, for example, produced commands to send to a satellite at some future time. The execution architecture referred to the time-critical component categories that facilitate communication of commands to and reception of telemetry from a satellite. Component categories across these two subsystems did not interact except by reading data from and writing data in shared files. The most common form of interaction would occur when an execution component reads (and transmits to the satellite) a command sequence produced by a planning component.

4.2.1.1 Execution Architecture

Figure 2 shows the data flow view of a product architecture using the CCT architecture's execution subsystem. The shaded area covers components provided by CCT. The arrows indicate control and/or data flow.



Operator Display
Operator Display
Operator Display
Operator Display
Timeline Display
Schedule Passplan, create connectivity
Maneuver Station keeping, prediction, momentum, mgmt, injection and transfer
Orbit Generation, estimation, event. analysis
Attitude Generation, estimation, event. analysis
Evaluation retrieve, trend
CCT
Vehicle Components, Control, Environment
Planning Data Access
ORB
Store
Persistence Service
Naming Service
Event Service

Figure 2: CCT Execution Architecture: Data Flow View

Variation is supported by the addition of the components outside the shading. The CCT components within the five component categories provided common features and also supported variation as follows.

Status Component Category

Status referred to the processes needed to receive telemetry data streams from either ground equipment or the spacecraft, perform integrity checks on the data stream, and decommutate the data into last recorded values (LRVs), which represent the latest and best estimate of the parameter's raw value. Variation points included how the status component would recognize and respond to format changes, as well as how the component would perform validity checks.

LRV Component Category

LRV received raw decommutated telemetry data from status processes. Processes in LRV then performed predefined conversion operations to generate engineering values for the decommutated data, performed limit checks on the data, generated alarms in response to undesired telemetry states, and provided LRV data to other processes. The logging of LRV data formed the key interface for providing data to the planning part of the system. A key variation point was the ease with which new LRVs could be defined and integrated into existing LRV processing. CCT provided LRV processing components that performed basic conversions, limit checking, and alarm generation and supported various forms of customization through parameters and inheritance.

- New algorithm definitions were allowed, and a distribution service was provided.
- Definition of new LRVs was supported.
- LRVs could be defined in terms of other LRVs, providing the ability to define higher-level state information.

Control Component Category

Processes in the control component category encoded client commands to external devices (ground equipment or spacecraft) into data packets or streams with appropriate formatting. CCT provided a command request interface to process and release regular and time-critical commands. Basic formatting algorithms could be parameterized or replaced. Unique verification logic could be added. CCT also provided a procedural control language that can be used to automate complex commanding sequences.

On-Board Processing Component Category

On-board processing referred to processes that model and help manage computer processor memory on board the spacecraft. Common capabilities included providing state information to clients and comparing the predicted memory map with telemetry-based snapshots of the actual memory map. CCT provided a memory map model with interfaces to support product extensions.

History Component Category

History referred to the processes that log and retrieve data received or generated during contact with a spacecraft. CCT provided common logging and retrieval from short-term storage to flat files. Product builders would be responsible for providing alternative persistence strategies, such as database logs and data replication.

4.2.1.2 Planning Architecture

Figure 3 shows the data flow view of a product architecture using the CCT architecture's planning subsystem. The shaded area covers components provided by CCT. The arrows indicate control and/or data flow.

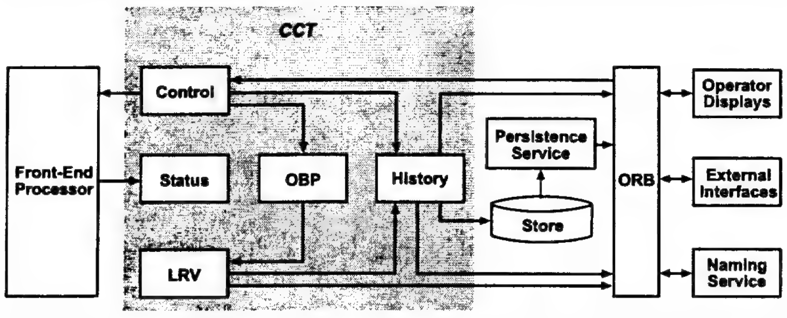


Figure 3: CCT Planning Architecture: Data Flow View

The CCT components within the six component categories of planning provided common features and supported variation as follows.

Orbit Component Category

Orbit referred to processes that estimate a spacecraft's orbit parameters based on observed data (state determination) and then propagate the orbit through time. CCT provided a standard family of estimators and propagators, including those commonly used by systems for spacecraft command and control missions. Variation points permitted product-specific tools to generate orbit data or use outside sources to provide orbit data.

Attitude Component Category

Attitude referred to processes that estimate a spacecraft's attitude (orientation) based on observed sensor data and then propagate the attitude through time to accomplish prediction. CCT provided attitude estimators and propagators for spin-stabilized and three-axis-stabilized spacecraft. Products could integrate their own tools to generate attitude data or sources to provide attitude data.

Maneuver Component Category

Maneuver referred to those processes that plan and generate maneuver sequences. These sequences use thruster (jet) firings to change a spacecraft's orbit, attitude, or momentum rate. Maneuver is key to orbit and attitude maintenance, taking orbit and attitude information, along with vehicle-specific propulsion system models, and generating the necessary plan to maintain or achieve the desired orbit or attitude. Maneuver planning depends heavily on the specifics of the spacecraft and its mission. Since the variation in this area is so wide, CCT provided general component frameworks for developing maneuver-planning processes. CCT implemented a common solution for spin-stabilized and three-axis-stabilized spacecraft. These implementations could be further generalized as the product line matures.

Vehicle Component Category

Vehicle referred to components that provide mission-unique spacecraft models to other components. The vehicle component captured several common interfaces of a spacecraft, with each parameter requiring user definition and implementation. These parameters included the following spacecraft characteristics:

- A unique identifier
- Mass properties as determined by the positions and mass properties of the spacecraft subsystems and appendages
- Orbit, attitude, and time state

The vehicle component category provided extensibility so that users could capture their mission-unique spacecraft features. Users could freely extend the given vehicle model for their spacecraft by defining new spacecraft components and component relations.

Schedule Component Category

Schedule referred to processes that plan spacecraft and ground system activities. The scheduling process results in a chronological set of scheduled activities from which plans are generated. The CCT architecture provided an extendable framework and a default implementation that accepted inputs through a timeline display, scheduled the activities, and then generated plans for a satellite pass. Variation points included selection (or addition) of scheduling and conflict-resolution algorithms as well as mission-unique requirements and data.

Evaluation Component Category

Evaluation referred to the processing of previously collected telemetry and control data. This processing could generate trending or unit history data or be used to play back archived data through the execution telemetry and control component categories. Evaluation was performed in support of the needs of component categories in both the execution and planning subsystems. Variation points included inserting new algorithms for controlling playback speeds and processing data.

4.2.1.3 Variation Summary

Within each of the CCT subsystems, use of individual components could vary, according to mission-unique requirements. Variation points were identified during domain specification, propagated through analysis and design, and supported by the CCT architecture. Depending on the type of variation point, CCT provided one of six standard mechanisms:

- Dynamic attributes
- Template
- Inheritance
- Parameterization
- Function extension (callbacks)
- Scripting

Table 2 provides a summary of the breadth of variation CCT can accommodate [Shaw 00].

Table 2: Component Variation Across CCT

Domain	Component Category	Components	Variation points
Execution	Status	4	1
	LRV	4	1
	Control	6	14
	On-board exec	3	2
	History	5	7
	Other (Playback)	1	1
Planning	Orbit	7	19
	Attitude	4	9
	Maneuver	5	12
	Vehicle	3	5
	Schedule	3	13
	Evaluation	4	3
Object services		10	13
Infrastructure		42	11
TOTALS		100	110

The process of starting with the architecture and reusable components and building specific applications included the following steps:

1. Determine COTS and legacy usage in the end system.
2. Identify real-time user interface products and database implementations.
3. Select a CORBA vendor to provide the intrasystem communication infrastructure.
4. Address security needs by adding security layers or secure gateways.
5. Determine how to extend and vary the CCT components by means of inheritance or the use of parameters. Replacement components may take the form of COTS products, legacy code, or hardware.
6. Package the components into executable applications and allocate them to the nodes of the end system's physical network.

4.2.2 Architecture Evaluation

Both the planning and execution architectures underwent explicit evaluation led by the SEI using the Software Architecture Analysis Method (SAAM) [Bass 98]. SAAM gathers together stakeholders of a system and lets them brainstorm scenarios of usage and modification that the software architecture of that system should support with little or no change. For CCT, SAAM produced scenarios of usage for ground-based spacecraft command and control systems. The scenarios that the CCT architecture supports conveyed an idea of the scope of the product line for which CCT should prove useful. Some of the scenarios were aimed at illuminating the production plan by which systems are built from the CCT architecture and components.

Neither the planning nor the execution architecture evaluations revealed any modifiability (extensibility, variability) problems with CCT. That is, no scenario adopted by the evaluation group revealed any problems that would require the CCT program or a user more than a few person-months to correct. Most scenarios either were already accommodated by CCT variation points or would require only minor changes to support.

For the planning subsystem architecture, the scenarios were as follows:

- A user wants to include mission management (payload planning, for example). How would CCT planning interact with and support mission management?
- The database is replaced with a new database technology for which the existing abstract interface is insufficient.
- The network and/or some computers go down, but users want the planning processes to continue operation.
- A user wants to manage a constellation of satellites (for example, global positioning system [GPS], communication satellite). Replace the vehicle model with an abstraction to represent a constellation.

- A user chooses to integrate another orbit package to replace one provided by CCT. More specifically, a user integrates a proprietary orbit package with different inputs (for instance, orbit packages are frequently optimized for a particular orbit).
- A user adds a new vehicle and associated "stuff" to existing system (for example, go from a spin-stabilized to a three-axis-stabilized vehicle). Use CCT to support up to six different vehicle families simultaneously from the same installation.
- Add continuous orbit management to component categories in the planning domain.
- How well will CCT support flexible scheduling using more on-board intelligence? Suppose some kind of request for service comes from the satellite to CCT, (for instance, a message that says a data buffer is full).
- A user requests testing of a legacy database for inclusion in CCT.
- How will a CCT component utilize a service that didn't exist when it was built?

For the execution subsystem architecture, the scenarios were as follows:

- Integrate an inference engine to perform constraint analysis.
- Inhibit commands.
- Change the computer platform, operating system, ORB, ORB version, operating system version, implementation language, or methodology. Change from a relational database management system to an object-oriented database management system.
- Add a new command source.
- Add telemetry-command synchronization (closed-loop control).
- A software routine called "via callback" hangs up (perhaps becomes stuck in an infinite loop). How does the system recover?
- The operator reconfigures a set of components dynamically.
- Add more automated, finer control to component categories in the execution domain.
- Increase the number of simultaneous telemetry streams.
- Send high-priority emergency commands to save a spacecraft (support command prioritization).
- Build a minimal execution application and add to it incrementally.
- A user evaluates a CCT component with respect to dependencies. A system architect assesses CCT architecture. An application architect assesses CCT status components.
- Have the telemetry stream update the on-board processor's memory map in real time.

Future users of the CCT assets would benefit from the evaluations and the increased attention paid to the architecture. The inclusion of software architecture evaluations in the CCT development process highlighted the need for a shared architectural vision; the creation of the Architecture Model and the Reuse Guide was a response to that need to document the architecture to make it more comprehensible to the stakeholders. The CCT architecture working group watched over the growth of the CCT architecture, and members continue to provide guidance during asset evolution.

CCT processes, as robust product line processes should be, are centered on the architecture. The Architecture Model and Reuse Guide emerged as focal points for the integration of the development processes for CCT assets and for target systems derived from those assets. The architecture focus for the development processes greatly improved the documentation of the architecture for both developers and users alike and did much to dispel the original notion that CCT is "just a toolkit."

4.3 Component Engineering

Component engineering produced the components within the various component categories. These categories were identified in the Domain Definition Document and specified in the Domain Specification Document. The reusable components created by component engineering constitute the development view of the "4+1 view" model. Component engineering was also responsible for unit testing of the components. There were two potentially conflicting goals related to component engineering:

1. The reuse of DCCS components: CCT sought to use components from this legacy system as a starting point.
2. The creation of reusable CCT components: CCT assets could not be limited to capabilities of any individual system.

Coupled with these goals, the CCT program needed to provide components to support the first real CCT user, the Spacecraft C2 System, and to do so in line with that project's schedule. They took a number of measures to achieve these goals. To the best of their ability, they strictly adhered to their development plan in the planned increments so that the Spacecraft C2 Project would not fall behind schedule. They built components that were capable of handling the variation points in the domain specifications and that met the interface specifications dictated by the architecture. DCCS code that was used was wrapped, reengineered, or thoroughly reviewed to ensure that it would work in the CCT context. The Spacecraft C2 team was involved in the review of all component development work. The Web access to all of the CCT artifacts gave Spacecraft C2 early visibility into the progress of the component development effort.

The components in each component category were implemented to have the interfaces and interconnection mechanisms called for by the software architecture. However, component interaction in products was still a concern: the interconnection mechanisms provided were so rich that product builders could easily assemble components into a configuration not envisioned by the CCT architects. To prevent this, the Reuse Guide established restrictions on mechanisms that product builders should use for component interaction. The descriptions of component interfaces were included in the Architecture Model. The Architecture Model also contained use-case maps to illustrate how components should interact during a scenario.

4.4 Testing: Application and Test Engineering

Application engineering and test engineering together built the system test architecture and executed detailed test procedures. The test engineers validated CCT reusability, including the architecture and other assets.

Recall that for CCT, application engineering was an internal reuse process that assembled a spacecraft command and control application from the reusable assets delivered by the other processes. For each CCT increment, this application was called a system test architecture. Test engineering used the system test architecture to perform the formal testing of CCT components. The formal testing complemented the informal testing done during component and application engineering.

CCT defined four levels of testing:

- Level 0: Unit testing of components (informal, performed by component engineering)
- Level 1: Requirements verification of components and component categories (formal, performed by test engineering)
- Level 2: Integration testing: The test architecture was used to test the integration of CCT into an application development effort and to exercise the variation points defined in the use cases (informal, performed by application engineering)
- Level 3: System-level performance requirements verification (formal, performed by test engineering)

Level 3 testing, in particular, was the formal test engineering effort used to demonstrate to both the quality assurance staff and the customer that system and performance requirements were met. Specific performance goals (for example, the ability of the architecture to handle some specified number of LRVs) were levied by satellite programs such as the Spacecraft C2 System.

The application engineering team was, in effect, the first user of the CCT assets. Their validation of the assets complemented the activities of the domain engineering group. For each defined CCT increment, the application engineering process attempted to create a "complete" spacecraft command and control system from the CCT assets. The testing process facilitated the exercise of the defined variation points and tested system-level requirements (such as, overall system-level runtime performance). Figure 4 represents the process of using assets first for building the system test architecture, then for building the first operational system.

After the testing of the assets created for that increment, the production plan provided the steps for producing the appropriate increment of the Spacecraft C2 System. The CCT approach repeated this step for each increment.

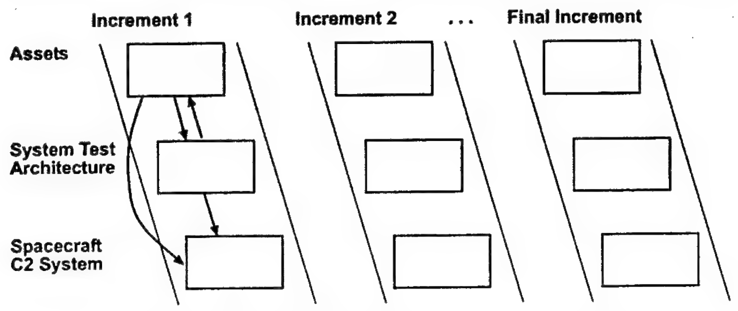


Figure 4: Production Plan for Building Test and Operational Systems from Assets

The Test and Integration Plan described the test methodology in generic terms for the first increment and then in more specific terms on a per-increment basis. The test engineering team created test plans and procedures based on the use cases in the Domain Specification and demonstrated the portability of the CCT assets across several platforms. They created test cases (high-level preliminary descriptions of how requirements were tested) and test procedures (elaboration of test cases in terms of specific inputs and steps) for the formal tests. The component engineering team created level 1 test drivers, whereas the application engineering team created the executable test system for level 3 testing. CCT used a discrepancy report process to document problems that occurred during testing. The discrepancy reports documented and tracked all problems discovered during testing. This process was documented in the Test and Integration Plan. As part of the testing process, peer groups reviewed and prioritized test cases and procedures.

The test engineering process validated CCT's reusability. The process integrated CCT assets into a spacecraft command and control application. It also exercised the variability of the components by adding mission-unique extensions to CCT just as a real-world user would. The test engineering group participated throughout the CCT development cycle and in all the standing meetings. There was a conscious effort to ensure that CCT testing for each increment was not an after-the-fact process.

4.5 Sustainment Engineering: Product Line Evolution

The sustainment engineering process included tasks to maintain and evolve the CCT assets. A sustainment engineering team was the primary point of contact after an increment was delivered. Their work began after formal testing and delivery of an increment. The input they received included the following:

- Requirements that could not be met during initial component development were communicated. The team oversaw the process to consider these as inputs for inclusion in later increments of CCT.
- Post-deployment problems (discrepancy reports) on assets. Specific CCT review boards determined the impact of proposed changes in the CCT software baseline and determined the action that the sustainment engineers should take.
- Issues raised by potential use of CCT assets by targeted users. Many of these issues were taken up by the Architecture Group, but were not addressed during the CCT development.

The CCT Sustainment Support Plan documented the process for sustainment engineering.

The sustainment engineering team continues to deal with discrepancy reports to this day. In this maintenance role, sustainment engineering is building a community of CCT users who depend on CCT for a significant portion of their control channel needs. Multiple satellite programs have different, and possibly conflicting, requests for modifications of the CCT baseline. The sustainment activity continues to determine which modifications to support and which modifications may affect the architecture.

4.6 Documentation

Because it created assets for reuse, the CCT program produced some documents that are not found in typical government procurement cycles. Documents that exist because CCT was a product line effort include the following, some of which have been mentioned above but are collected here to highlight the special role documentation plays with software product lines.

- **CCT Domain Definition**: This document described the boundaries of the command and control product line and the functions that CCT addresses. The Domain Definition also included a glossary of terms and acronyms used within the CCT products.
- **Generalized Requirements Specification**: This document captured common capabilities to be provided by the toolkit. It was produced by examining requirements from three different customer bases: DCCS, the Spacecraft C2 System, and SSCS.

- **CCT Domain Specification**: This document provided a requirements analysis, employing use-case-driven methodologies and documented using UML.
- **CCT System Test Architecture**: This document described a test system architecture (design and implementation) used to verify CCT functionality. Users could use this document as a starting point for their designs, as an ad hoc implementation for factory testing, and as an example of CCT component integration.
- **CCT Portability Demonstration Plan**: This document contained an assessment of portability to selected non-supported operating systems.
- **CCT Sustainment Support Plan**: This document detailed maintenance and evolution management for the CCT assets. Users who wish active involvement in the CCT's management would participate through the described plan.
- **CCT Program Concept of Operations**: This document described the life cycle of the CCT product line, the stakeholders who have an interest in the product line over its lifetime, and business concerns such as qualifying a new subscriber (user) for the product line.
- **CCT Architecture Model**: This model provided a complete description of the architecture. Future product developers use it to evaluate CCT capabilities with respect to target program requirements, to determine how to plan their use of CCT assets, and to evaluate what changes and extensions they need to provide.
- **CCT Reuse Guide**: This document described and provided examples of the steps necessary to build a product line application from CCT assets. It was the CCT production plan. The CCT Reuse Guide did not prescribe a development process, but rather provided the process basis for reuse analysis and design. Users could determine how best to incorporate this basis into their own development standards and practices.

5 Managing the CCT Effort

The management of a product line effort, especially when the effort marks a departure from the standard approach to software development, requires sophisticated management skills and well-honed leadership qualities. Clements and Northrop lay out a healthy collection of technical management practice areas and an even larger collection of organizational management practice areas [Clements 01]. We have already described the CCT practices for a good number of these areas. The technical management practices and the management style are most notable.

First, we focus on technical management. Raytheon's process sophistication and discipline resulted in superb execution of CCT's technical management activities. Processes for technical planning, configuration management, technical risk management, process definition, data collection, metrics and tracking, make/buy/mine/commission analysis, scoping, and testing were stitched into the fabric of the CCT development process. There were very few process snags throughout the entire CCT development process, and none that weren't handled—which brings us to the management style of those at the helm of CCT.

During the feasibility study prior to the CCT official launch, the entire NRO management chain responsible for CCT supported strategic reuse. They could all be described as visionaries. They all recognized the unnecessary duplication and risk involved in stand-alone efforts for similar spacecraft systems. They all were open to the changes that a product line approach would necessitate in terms of the organizational and technical practices to which they were accustomed. They stayed the course and provided unwavering support.[1]

In particular, the CCT Program Office manager, John Ohlinger, was deeply committed to the success of a product line approach. He maintained a loose, relaxed management style, providing sufficient direction while being flexible enough to hear ideas, bring in experts, and respond midstream to perceived needs. For example, early on he became convinced of the need for a greater focus on the architecture in order for CCT to succeed. He altered the original plan to incorporate two architecture evaluations despite some dissention within his office and the contractor team. He then formed the CCT Architecture Group to address the evaluation findings and to continue to shine a light on architectural issues and their resolution.

[1] Near the completion of the CCT effort, those in the NRO chain of command above the CCT Program Office moved on, one by one. As others filled their ranks, the support for the effort diminished.

His counterpart, Jeff Shaw, Raytheon's CCT program manager, was a high-energy individual and a firm believer not only in software product lines but also in the ability of his team to deliver the CCT asset base. His enthusiasm was contagious. His support was obvious in the way he treated his staff and handled their needs. He was structured and well organized, a superb communicator who made others in his organization into CCT believers. Both he and Ohlinger had a great deal of technical savvy and experience in the spacecraft command and control domain. Ohlinger's and Shaw's styles were complementary; together they provided a protective shield and a supportive environment for the CCT development team. They didn't just manage; they led.

Close to CCT completion, Jeff Shaw was reassigned. His deputy assumed his responsibilities and applied her own leadership skills to complete the CCT effort on time and within budget.

6 Early Benefits from CCT

The CCT story continues to unfold; user systems are still in development and the CCT sustainment processes are still relatively immature. Nonetheless there are early benefits to herald. First and foremost, CCT is a ground-breaking effort within its government communities. It broke down organizational and cultural barriers to garner support for a product line approach, and it achieved its objectives. CCT was completed on schedule and within budget in December 1999 with no outstanding risks and no outstanding actions. Few software efforts in its class can boast such a track record. Moreover, CCT is being used successfully to build the Spacecraft C2 System.

The initial NRO business case for pursuing a product line approach sought benefits in several areas. These included:

- reduced development costs
- increased quality in the form of continuous improvement and evolution
- decreased time-to-field
- savings in sustainment costs

This business case attempted to quantify cost savings over a ten-year life cycle, including CCT development, use, and sustainment. The NRO determined that costs during the period of intense CCT development (1997–2000) would be higher than normal, in spite of near concurrent use with the Spacecraft C2 System. However, when forecasting was done over the longer term (1997–2009), it was determined that overall cost savings would accrue for systems using CCT as follows:

Development	18.2% savings in anticipated total government development-related expenditures
Sustainment	27.8% savings in anticipated total government sustainment-related expenditures

Savings in both cases are in comparison with expected costs using a non-product line approach.

6.1 First CCT Product

The Department of Defense has already enjoyed specific, measurable benefits in the initial use of CCT on an actual operational system—namely, Spacecraft C2. Because of CCT, the Spacecraft C2 Program is enjoying reductions in development costs, schedules, work force, and product risk, as well as increased product flexibility. Table 3 summarizes these benefits.

Table 3: Summary of Measurable Benefits Attributed to Use of CCT

Factor	Benefits to Spacecraft C2 System
Quality	One-tenth the typical number of discrepancy (defect) reports for a system of this type. The problems identified were all local ones, with localized fixes, having no ripple effects, and no effect on the architecture.
Performance	Use of CCT improved performance over results predicted without CCT. In identified places where reuse may lead to timing problems, CCT variation points can be exercised to apply mechanisms that circumvent CCT software and apply faster algorithms.
Integration time	Incremental builds completed in weeks rather than months, as was the case for non-CCT portion. This approach is a direct carryover from the incremental approach to development.
Code volume	The number of design objects for subsystems using CCT is lower than planned by 34% to 88% with similar reduction in actual source code size. Total SLOC developed by Spacecraft C2 is 76% less than planned.
Productivity	Smaller development staff required (15 versus 100 for other similar systems) Overall costs cut by 50% Overall schedule cut by 50% Documented flexibility in meeting requests for modifications by the Spacecraft C2 System customer CCT treated like a COTS product (initial training required, then development proceeded on the basis of domain specification, interface definitions, and Reuse Guide)

Interviews with the Spacecraft C2 developers also revealed other, less tangible benefits. Program staff attrition is unusually low, at two staff members in three years. After initial training, these developers were very satisfied with the CCT approach and praised the selection of and support for the variation points. They reported greater professional satisfaction with the product line approach. They expressed a sense that the pedestrian tasks had already been done and the focus was on the interesting, mission-specific capabilities. The risks of system failure were felt to be fewer, and the tension on the program to be substantially lower.

6.2 Benefits Beyond CCT Products

The government and Raytheon have achieved the intended benefits from CCT. Spacecraft C2 has achieved measurable benefits, and there are other users beyond Spacecraft C2. The government has already met three of its first four goals: reduced development costs, increased quality, and decreased time-to-field. It is too soon to calculate the savings in sustainment costs.

Raytheon is also capitalizing on the CCT experience by using the CCT assets in its other systems and by using the CCT processes and tools on other efforts.

Other commercial organizations will have access to CCT products and may use similar approaches for launching their own product lines in spacecraft command and control. The Software Object Technology Group, a joint government and industry group led by NASA and the NRO, has applied the CCT architectural concepts to the definition of an object/interface standard for space-related applications.

7 Lessons and Issues

With asset development complete, CCT offers its users a set of tested assets to support development of new ground-based spacecraft command and control systems. Many lessons were learned during the CCT effort, and several major challenges still face the CCT program as it evolves from an effort primarily focused on asset development to one dedicated to product development by external organizations. The use of CCT for development of the system test architecture and the Spacecraft C2 System brought many nagging issues to the surface. Future development and collaboration with external users will shed light on others. We conclude the CCT case study with a brief tour through some of these lessons and outstanding issues.

Tool Support Is Inadequate

Tools for adequate support of software product lines lag behind the adoption of product line approaches, and so it's no surprise that the tools for support of CCT processes were not ideal.

In common with many domain analysis efforts, the CCT domain analysis used existing object-oriented analysis and design tools to capture and represent domain-level concepts. The outputs of the domain analysis process were spread over a number of documents that incorporate text, outputs from the Rational Rose tool, and outputs from the Slate tool. Collectively, these documents comprise the requirements database for CCT; there is no single, easily maintainable model. Maintenance is painful, which may pose a risk for long-term use of the assets. The same is true for the Architecture Model, which contains the layer model diagrams, use-case maps, and overall execution and planning architecture diagrams. Owing to a lack of tool support, these diagrams exist separately. The distribution makes maintenance awkward and time-consuming and increases the risk that changes will not be incorporated consistently.

Lack of consistent tool support for traceability among the CCT assets continues to hamper ongoing CCT maintenance. The CCT team documented a complete set of generic requirement specifications based on the initial analysis and subsequent understanding achieved through design and implementation. A formal, tool-supported mechanism should record issues and decisions, and feed them back into the domain model and other analysis results. Unfortunately, the tool support doesn't yet exist, and the resultant manual efforts are cumbersome.

Domain Analysis Documentation Is Important

The CCT domain analysis products did not contain a record of issues and decisions regarding the presence or absence of particular domain features, or a record of tradeoffs that might influence subsequent design decisions. Such information would have been useful to CCT developers. It would also be useful now to targeted CCT users as they try to decide whether or not to use CCT for their products and to determine how CCT could be used.

An Early Architecture Focus Is Best

A software product line approach relies on a strong product line architecture that is used to structure all of the products in the product line. The architects need to take great care to ensure that the product line architecture is accessible to and understandable by all future product builders. Because of the acquisition nature of this product line effort, even greater care was needed; the product builders will not be members of the CCT development team and probably will not belong to the Raytheon Company, nor will the CCT Program Office commission them. In order for the NRO to benefit fully from the product line that CCT is intended to support, the CCT architecture documentation and its attached process must be of exceptionally high quality. Moreover, the NRO would like some means to ensure that product architectures (called target architectures by the CCT effort) actually conform to the CCT architecture. They want to ensure that the CCT assets are used in the intended way and not simply to feed some ad hoc reuse of components that would undermine the true potential of CCT.

Unfortunately, the CCT effort was not architecture-centric from the start. Remember that the original vision called for a toolkit. The architecture and architectural knowledge were in the heads of the CCT architects, who did a fine job and who remained involved in every aspect of the CCT development and its initial use. Although this involvement was a plus, it didn't really permit the architecture documentation to be exercised to the fullest by the development team. Team members could always go ask the architects, and for that matter, so could the first product builders.

Early on, the SEI was concerned about the architecture and its documentation and advocated two architecture evaluations and the development of the Reuse Guide. The CCT Program adopted these suggestions but had to insert these efforts into the schedule after the project was well under way. It is to the Raytheon team's credit that they embraced both activities even though architecture evaluations weren't in the original contract.

During the first software architecture evaluation, it was apparent that there was a lack of architecture documentation that could stand on its own. The CCT Architecture Group was established to address this need. This group advocated creation of the Architecture Model and was a strong proponent of the Reuse Guide. Both documents attempt to communicate CCT architectural concepts and decisions to CCT users who would build future products. These documents, however, were not in the original plans and suffer a bit from their late start. They are not as user-friendly as they should be, and in their current format it is more difficult to determine architecture conformance.

Product Builders Need More Support

The CCT effort focused on the creation of a set of assets for spacecraft command and control and much less on the integration of those assets. There were two outcomes that have proven to be disadvantageous to future product builders:

1. Crafting an end-to-end production plan, which starts with product requirements and ends with a plan for system integration, was never part of the CCT agenda.
2. More emphasis was placed on evaluating the functionality of the assets, in stand-alone fashion, than on testing their integration.

The Reuse Guide is the CCT production plan, and its very existence is positive. It provides extensive documentation on the use of individual component categories, their variation points, mechanisms, and tailoring—what we might call the attached process for components and what they call component categories. However, other information is less well organized and in some cases missing. Critical architectural information regarding the range of overall qualities (for example, runtime performance and reliability) of the systems built from CCT assets is either difficult to locate or missing. There really is no attached process for creating an instance of the architecture and no attached process for system integration and testing. A decision was made to leave the process up to the builder, which might be fine, but a product builder wants to get a system perspective first, and the Reuse Guide offers no direct support for such a perspective. The product builder has to glean this in a bottom-up fashion. The Reuse Guide is also missing any sort of CCT primer that explains basic concepts and assumptions and helps a potential user mount the CCT learning curve.

Production capability was tested in two ways: by having an internal application engineering team (the system test architecture developers) act as early users of CCT, and by having the first real user—the Spacecraft C2 System—cooperate in the CCT development effort. Both were positives for CCT. However, the end-system perspective wasn't pervasive enough during the creation of the test system, despite the involvement of domain experts and architects: the test cases were in some instances too simple to address the kinds of issues involved in integrating CCT assets into the development of an operational system. Integration and other system-wide issues, such as meeting a range of real-time performance requirements, were not adequately tested. The CCT component level requirements were met, but it was not entirely clear from integration-level testing that the needs of real, full-up command and control applications would be met. The first user was successful, but was intimately involved in the entire

CCT development process and so is not a typical future user, who will need more support. These key reuse aspects were not addressed early in the program to the same extent as were the technical requirements [Ohlinger 00].

CCT Users Need Reuse Metrics

There are two types of CCT users: those who would commission systems to be built from CCT (acquirers) and the contractors who would be doing the building. The acquirers are interested in knowing that the systems built by their contractors do in fact use the CCT architecture and do in fact take advantage of CCT as a product line asset base. They want to measure architecture conformance. The CCT Architecture Group developed conformance guidelines to support this user need. The conformance guidelines define six conformance levels. The Architecture Group applied these guidelines to the system test architecture to determine its level of conformance. The SEI conducted a conformance analysis of the system test architecture and the CCT architecture, at the component category level. Both results provided useful metrics, and also led to some recommendations on architecture enhancement.

There was no metrics program that identified specific measures for analyzing the reusability of the CCT assets or the benefits that would accrue from their reuse. Full tracking of the extent and remediation of defects provided some measures, but these measures were not preplanned. Since CCT completion, there has been an effort to collect data that speak to the reuse potential and the inherent benefit of using the CCT assets. These benefits were reported earlier and are now included in a business case being developed by the NRO. Such metrics are essential for attracting new users (acquirers) and convincing their contractors of CCT's merits; they both need to know the benefits and the costs in quantitative terms. Data should have been proactively collected from the beginning. There were potential users who rejected CCT because of the lack of such quantitative information.

It Pays to Be Flexible, and Cross-Unit Teams Work

True to its name, CCT was conceived as a "toolkit" development effort. The increments were created on the basis of staged delivery of component categories. The incoming assumption was that a rich set of highly flexible class categories could support the requirements of future spacecraft command and control systems. Significant effort was applied to understanding and documenting the variation points for support of variability and to the mechanisms for implementing the variation points, but the focus was on the functionality and availability of components.

The shift to an architecture-based approach came about as a result of the first architecture evaluation and a subsequent decision made by John Ohlinger and other key CCT players to form the CCT Architecture Group. The Architecture Group had representatives from the entire spectrum of major players in CCT development, including the contractor, technical support consultants to the CCT Program Office, the SEI, the Aerospace Corporation, and end-user organizations. Its chartered purpose was to guide the evolution of the CCT product line architecture, and its functions were to:

- investigate and assess the architectural impacts of design issues and new technology insertion
- oversee any proposed inclusion of major enhancements in the CCT
- promote the interoperability of CCT components with other component environments, commercial products, platforms, and other customer architectural environments
- promote the extensibility of CCT components by recommending architectural solutions that are reusable and customizable
- seek reductions in life-cycle costs through improvements in technology, methodology, specifications, and tools

The Architecture Group was the major influence in moving CCT from a component-based effort to an architecture-based effort. The lesson that was learned is really twofold: (1) structuring the organization is an ongoing process that requires organizational flexibility, and (2) a permanent crosscutting architecture group has great benefits.

A Real Product Is a Benefit

The Spacecraft C2 System was really the first to test the reusability of CCT. This program consisted of two major subsystems:

- Spacecraft command and control partially built using CCT
- Payload operations entirely independent of CCT

The program integrated CCT assets into its spacecraft command and control application, and it also exercised the variability of the components by adding mission-unique extensions. The Spacecraft C2 System supported CCT by providing essential feedback in two areas:

- **Test engineering**: The program evaluated system tests.
- **Sustainment Engineering**: The program provided feedback on actual component use.

Spacecraft C2 was a major influence on CCT and a factor in CCT evolution since the initial increment. This real product demonstrated the strategic intent of CCT—to show that satellite programs could use CCT as the basis for their spacecraft command and control solutions—and that demonstration was critical. Here is the lesson for organizations that commission a product line asset base: commission or develop a product in concert with the asset base development. The schedules and the demands of the product team have to be managed carefully, but the advantage of having a real product to provide valuable feedback and prove the usability of the assets outweighs any extra management required.

8 Summary

The development of the Control Channel Toolkit provides an illuminating software product line case study, an exciting advancement in satellite software development, and a major stride for software product lines within the U.S. Government. The NRO established both the product line assets and an organization to support and sustain their use, and they succeeded in their goals. The following themes, which pervade other successful product line efforts, [Clements 01] were echoed in the CCT story.

- Deep domain experience resident at Hughes/Raytheon and at the NRO
- A legacy base from which to build
- Process maturity
- A dedicated champion (in this case, two) throughout asset development
- Strong architectural vision
- An incremental development and refinement approach

CCT is still in its infancy. The CCT Concept of Operations calls for multiyear phases after development to improve both the assets and the process as they are applied in satellite programs. In spite of its relative immaturity, the accomplishments of CCT should provide guidance to other organizations intending to field software product lines. Figure 5 shows the expected growth in the application of CCT in government and industry.

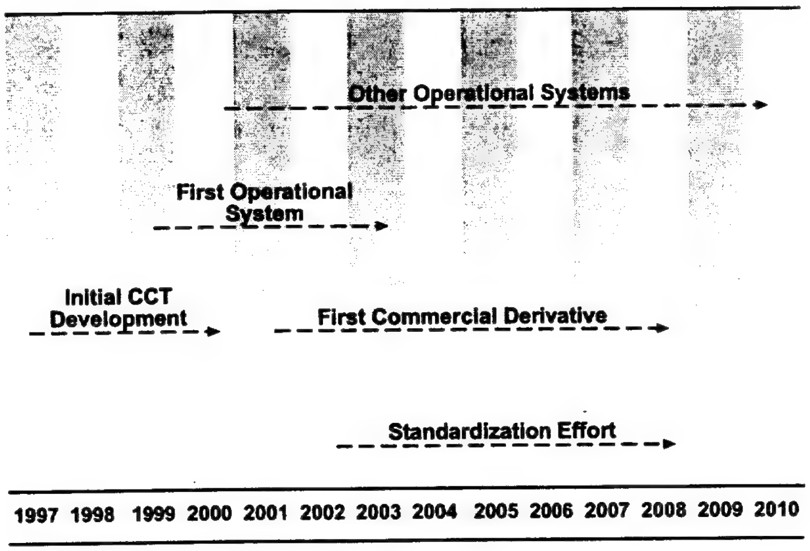


Figure 5: Growth in Use of CCT Assets

References

[Bass 98] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*. Reading, Ma.: Addison-Wesley, 1998.

[Brownsword 96] Brownsword, L. & Clements, P. *A Case Study in Successful Product Line Development* (CMU/SEI-96-TR-016, ADA315802). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.016.html> (1996).

[Clements 00] Clements, P. & Northrop, L. *A Framework for Software Product Line Practice, Version 3.0*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, September 2000. Available WWW: <URL: http://www.sei.cmu.edu/plp/framework.html> (2000).

[Clements 01] Clements, P. & Northrop, L. *Software Product Lines Practices and Patterns*. Reading, Ma.: Addison-Wesley, 2001.

[Hollander 99] Hollander, C. & Ohlinger, J. "CCT: A Component-Based Product Line Architecture for Satellite-Based Command and Control Systems." *Proceedings of Workshop on Object Technology for Product Line Architectures*. Lisbon, Portugal, June 15, 1999. Bilbao, Spain: European Software Institute, 1999.

[Jacobson 97] Jacobson, I.; Griss, M.; & Jonsson, P. *Software Reuse: Architecture, Process, and Organization for Business Success*. New York, NY: Addison-Wesley, 1997.

[Kruchten 98] Kruchten, P. *The Rational Unified Process: An Introduction*. Reading, Ma.: Addison-Wesley, 1998.

[Ohlinger 00]	Ohlinger, J. "CCT Lessons Learned What We Did, Why We Did It, and What We Would Do Differently." Presentation: GSAW 2000. Available WWW: <URL: http://sunset.usc.edu/GSAW/GSAW2000/pdf/Ohlinger.pdf> (2000).
[Shaw 00]	Shaw, G. "Control Channel Toolkit: Open Architecture-Based Product Line Development." Presentation: GSAW 2000. Available WWW: <URL: http://sunset.usc.edu/GSAW/GSAW2000/pdf/shaw.pdf> (2000).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE September 2001	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Control Channel Toolkit: A Software Product Line Case Study		5. FUNDING NUMBERS F19628-00-C-0003
6. AUTHOR(S) Paul Clements, Sholom Cohen, Patrick Donohoe, Linda Northrop		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2001-TR-030
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2001-030
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

This report is a case study of the Control Channel Toolkit (CCT), a software asset base for a software product line of ground-based spacecraft command and control systems built under the direction of the United States National Reconnaissance Office (NRO). Beginning with a characterization of the CCT context and a narration of the history of the effort, the report describes the management and software engineering practices, the software artifacts that were developed, the results that were achieved, and the lessons that were learned. It concludes with an accounting of the measurable benefits the government has already reaped in the initial use of CCT on a specific spacecraft command and control system. With the permission of Addison-Wesley, this report is extracted from *Software Product Lines: Practices and Patterns* [Clements 01], where it was published with the approval of the National Reconnaissance Office.

14. SUBJECT TERMS Control Channel Toolkit, CCT, product line, management practices, software engineering practices			15. NUMBER OF PAGES 64
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL